23rd International Conference on Knowledge Based and Intelligent Information and Engineering Systems

Towards A Knowledge Base for Software Developers to Choose Suitable Traceability Techniques

Haruhiko Kaiya\textsuperscript{a}, Atsuo Hazeyama\textsuperscript{b}, Shinpei Ogata\textsuperscript{c}, Takao Okubo\textsuperscript{d}, Nobukazu Yoshioka\textsuperscript{e}, Hironori Washizaki\textsuperscript{f}

\textsuperscript{a}Kanagawa University, Hiratsuka 259-1293, Japan
\textsuperscript{b}Tokyo Gakugei University, Tokyo 84-8501, Japan
\textsuperscript{c}Shinshu University, Nagano 380-0928, Japan
\textsuperscript{d}ISEC, Yokohama 221-0835, Japan
\textsuperscript{e}NII, Tokyo 100-0003, Japan
\textsuperscript{f}Waseda University, Tokyo 169-8555, Japan

Abstract

Huge amount of techniques for creating, maintaining and/or recovering traceability among software development artifacts have been proposed. It is thus not easy for a potential user of such techniques to choose a technique suitable for his/her project because too many techniques exist and projects are different from each other. In this paper, we proposed a model for characterizing a traceability technique with respect to its users. The model can become a meta-model of the knowledge base for such users. The model is designed on the basis of the contents of existing technical papers so that we can easily describe model instances on the basis of technical papers. The model is represented in a feature model, and it has four mandatory features: source, destination, consequence and process. The user can at least understand a technique is unsuitable for him/her by referring such features. If a technique estimates the traceability relationships, the instance of its feature model may contain the quality metrics of its estimation such as precision and recall. It has also several optional features: assumptions, preprocess and tool. Although we sometimes cannot obtain such optional features from technical papers, they are so helpful for a user to decide a technique suitable for him/her.

On the basis of the feature model, we described model instances of several techniques in technical papers. We also examined and discussed who the suitable user for each technique is.

© 2019 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of KES International.

Keywords: Software Traceability, Feature Model, User Perspective, Knowledge Base, Survey
1. Introduction

Introducing a software traceability technique in a software development project is crucial for many reasons\(^1\). For example, most software products are developed based on the previous version of each product. To know the impact on design and source codes by requirements changes, traceability relationships are required. All software products today are very huge. Even a small command such as info-zip contains 60 thousands lines of codes. Some operating system consists of more than 100 million lines of codes. To understand such huge products, relationships among the same level of products such as among codes are crucial. Relationships among different levels such as requirements and codes are also crucial. Traceability techniques contribute to improving such understandability even if the product has only one version. Traceability techniques also contribute to checking completeness and correctness of a product and finding reusable components in a product.

Because of the importance of traceability techniques, huge amount of papers have been published and consumed by researchers. In most papers, comparison between a proposed technique and others is shown, and the superiority of the proposed one is loudly presented in certain criteria. Several benchmark problems\(^2\) are shared among researchers, and used for such comparison. However, primal users of traceability techniques, i.e. software developers, are not interested in such feeble improvement. They want to know which technique is valuable and suitable for their projects. Projects are different from each other. For example, a rigorous model driven development is performed in a project. On the other hand in another project, artifacts such as requirements and design documents are written in natural language, and their changes are often notified and shared by email.

According to the background mentioned above, we setup the following research questions (RQs).

- RQ1: What kinds of information about a traceability technique are required for its user to decide whether it is suitable for his/her project?
- RQ2: What kinds of such information can be obtained from published documents such as technical papers?

On the basis of our survey\(^1\), we propose a feature model of a traceability technique as answers of the RQs. The model can be a basis of the knowledge base for software developers to choose traceability techniques suitable for their project. Although there already exists some frameworks for categorizing traceability techniques\(^1,3,4\), they do not intend to help users of the techniques to choose suitable ones. Such frameworks usually intend to improve and progress the research activities.

The rest of this paper is organized as follows. In the next section, related works are briefly reviewed and discussed. On the basis of papers in our previous survey\(^1\), we introduce a feature model of traceability techniques in section 3. Because the contents of technical papers are limited, we choose features acquirable from such papers. In section 4, several model instances based on actual papers are shown. Finally, we summarize our current results and show the future issues.

2. Related Work

Plenty of techniques for maintaining, developing and recovering traceability relationships among software artifacts are proposed in huge number of technical papers. Their survey have been also summarized in several papers\(^1,3,4,5,6\). However, few papers focus on how to choose suitable technique(s) for each user. In a paper\(^5\), understanding stakeholders and traceability strategizing techniques are focused as future research directions. Although both directions are slightly related to traceability selection, the paper did not mentioned such relationships at least. In a survey\(^7\), usage scenarios of requirements traceability are summarized. However, how to choose the technique in each usage is not investigated. Bangchao Wang et al\(^8,9\) focus on technology transfer from academy to industry. They proposed a maturity model of a technology whether it can be transferred to industry. The model contains many aspects of a technology such as usability, usefulness, customizability, quality, risk and cost. Although the model seems to be reasonable, it is not realistic to gather such many aspects only from technical papers related to a technology.
3. A Feature Model of Traceability Techniques

According to papers mentioned in our survey, we develop a feature model of traceability techniques in Figure 1. As mentioned in introduction, there is no frameworks for software developers to choose a traceability technique so that they try to use it. Some kind of knowledge base of traceability techniques will contribute to helping such developers. The model in Figure 1 is designed so that the model can become a meta-model of such knowledge base.

Before explaining each feature, we explain the criteria of selecting features in the model. We also explain the difference between mandatory and optional features in this model. To help software developers to choose their technique easily and suitably, necessary information for using each technique should be provided as much as possible. Features in the model are chosen on the basis of such criteria. Instead, technical details how to establish traceability are intentionally excluded from this model. For example, we regard kinds of information retrieval methods such as Support Vector Machine (SVM) or Latent Semantic Indexing (LSI) are not necessary in this model. We regard developers cannot choose suitable technique without mandatory features in the model.

3.1. Source (Mandatory)

A traceability technique creates relationships between parts of software development elements. Examples of such parts are a paragraph in a requirements specification, a Java class, a feature of a software product line, and UML model elements. In some case, a member of a project, i.e. people, can be a source of a traceability technique. A model of the technique thus has to specify one of such parts.

3.2. Destination (Mandatory)

The feature “destination” specifies another end of a traceability relationship. In many cases, it is not clearly specified that a traceability relationship has direction. Once a directional traceability relationship is established, its reverse direction can be easily added. Therefore, we do not distinguish the source and destination features. These features simply specify one end and another of a traceability relationship.

3.3. Meaning

A traceability relationship creating and maintaining a technique sometimes has its specific usage. Suppose a feature in a software product line is implemented as a class of Java, and a technique maintains such relationships between
features and codes\textsuperscript{14}. In such a case, the user of the technique can identify features which are not implemented in codes of a specific application. We regard the meaning of the traceability is “implement” in our model. Other examples of the meaning are “refinement”, “design rationale” and “impact”. Kinds of traceability links such as overlap, satisfiability, conflict and dependency are enumerated in a handbook\textsuperscript{15}. Such links types are useful to specify the meaning. When such meaning is specified, potential users of the technique can easily examine and choose the technique. According to a survey\textsuperscript{7}, about 80\% of projects applied traceability techniques because they expect their benefits. The feature meaning clarifies such benefits to users. However, in several papers, such meaning is not clearly specified in each technique\textsuperscript{1}.

3.4. Assumption

There is no general and standard software development project. Each traceability technique thus has several assumptions about a development project where the technique is used. When a user examine and choose existing traceability techniques, these assumptions are very important. If a project of the user does not satisfy such assumptions, the technique cannot be applied to the project. Otherwise, the project has to be modified so that the technique can be applied. The assumptions thus help users to estimate potential costs of introducing the technique.

Examples of such assumptions are as follows. In some techniques, information retrieval techniques are applied to documents, and many of them assume the documents are written in English\textsuperscript{16}. The naming convention of classes is sometimes assumed\textsuperscript{11}. Several techniques\textsuperscript{17,18,19,10} require software development history or logs. A technique\textsuperscript{20} assumes domain ontology that are used for semantic matching among words and terms. Development styles such as software product line\textsuperscript{12,21} and model driven development\textsuperscript{13} are also assumed for each technique. When traceability relationships are directly and manually created and maintained, some additional tasks are added to its software development. Such additional tasks are also kinds of assumptions. For example, using some domain specific language\textsuperscript{22} is a kind of assumptions.

Scalability of a traceability technique is important in general\textsuperscript{9}. However, general descriptions or metrics of scalability do not help users to choose a technique. Instead, assumptions exemplified above let users know whether a technique has a certain level of scalability suitable for them.

3.5. Consequence (Mandatory)

Each technique recovers or develops traceability relationships. If a technique recovers the relationships on the basis of a probabilistic approach such as information retrieval methods, not all relationships are correct in general. On the other hand, all the relationships are basically correct if a technique enforces some notations and tasks to maintain the relationships. For users of the technique, we have to clarify whether a technique is first or last type. We call the first type as “estimate”, and the second on as “determinate”.

For the first type, we also have to specify the quality of estimated relationships. Following two metrics are useful to know the quality of a technique.
Note that $|\text{relationships}|$ is the number of relationships. Composite metrics such as F-measure are not useful for people who want to choose a traceability technique because it does not let them know two different criteria respectively; one is how much wrong relationships are contained in the results by the technique and another is how much right relationships are missing by the technique. The tradeoffs between precision and recall are also very useful for technique choosers. In some research, some parameter is used and changes of recall and precision values are shown along the progress of the parameter as shown in Figure 2. Precision/recall curves as exemplified in Figure 3 are more useful than the changes of precision and recall respectively because users can directly know the tradeoffs between precision and recall. Because metrics such as precision and recall are calculated on the basis of some data such as benchmark examples, the values do not show the quality of a technique in general. Therefore, it is better to explain the characteristics of such data, e.g. size, complexity and artifacts in a class room or an actual project.

In the context of writing technical papers, comparing a technique to others are very important. However, such kind of comparison is not so important in the context of using techniques because the absolute level of the quality is important. For example, it is meaningless for users that the recall of a technique becomes 1.2 times better than existing one if the absolute values of the recall are only 10% and 12%. It is too terrible that about 80% of correct relationships are missing by the technique.

### 3.6. Preprocess

In the case of estimated techniques, several preprocessing tasks are required before recovering traceability relationships. Such tasks are normal negative factors of introducing a technique because software developers have to perform additional tasks. Although some preprocessing tasks can be performed automatically, such automation normally negatively affects the quality of the preprocessed data. Especially for textual artifacts such requirements documents, such automation is not so easy. For example, tasks such as sectioning documents, unifying terms, removing general terms are typically required and they are not fully automated. Other types of preprocessing are setting the list of keywords and also setting some value as the threshold. There is a tradeoff between the amount of preprocessing tasks and the degree of processing automation.

In the case of determinate techniques, there are few preprocessing tasks because tasks for creating and maintaining traceability relationships are embedded in the software development process itself. In our model, such issue is noted in the “assumption” feature mentioned above.
3.7. Process (Mandatory)

In each technique, several tasks are performed for creating, maintaining and/or recovering traceability relationships. The user of such techniques is interested in what and how many additional operations he/she has to perform. We call a technique performs a “batch” process if the user performs nothing for traceability during the process. On the other hand, a technique performs an “interactive” process, and its user has to do something during the process. A typical operation in an interactive process is user’s judgement. In a technique\(^9\), users give their judgement of classified traceability relationships.

3.8. Tool

If tools for performing a traceability technique exist, the number of its potential users increases. We thus specify the status of such tools for each technique. With respect to tools, we categorize techniques into the following three types: none, parts, integrated. If a technique or its papers state there is no tool support, we categorize the technique is “none”. If they do not introduce or do not mention any tools, this feature is not specified. When some pieces of tools are used in a technique, we categorized it is “parts”. For example, Java packages for vector space model, LSI and Sufficient Dimensionality Reduction (SDR) implementations in MATLAB are used in a technique\(^{16}\). Several techniques provide integrated tools to perform them. In such cases, we call a technique is “integrated”. Some tools are stand alone\(^{12}\), and others are built on a well-known IDE such as Eclipse\(^{20,25}\).

4. Model Instances

In this section, we describe several model instances of traceability techniques on the basis of existing technical papers. We also discuss for what kind of users each technique is suitable.

4.1. Information Retrieval based Traceability Recovery using User Feedback

In a paper\(^{29}\), traceability between documents are recovered on the basis of their textual similarity.

- Source: document
- Destination: document
- Assumption: Artifact types such as requirements, design, test and codes are identified.
- Consequence: Estimate.
  - Precision: Yes.
  - Recall: Yes.
  - Tradeoff: Yes, in the same way as shown in Figure 3.

Data of following three projects were used to calculate them: Easy-Clinic, iTrust, Moderate Resolution Imaging Spectroradiometer (MODIS). First two projects are contents for university education. The last one is an open source software by NASA. Project Web pages introduced in this paper do not exist in March 2019.

- Process: Interactive. A user has to judge whether first candidate of a traceability relationship is correct or not repeatedly.

If a user can perform the judgement mentioned in this technique, this technique is suitable for him/her. Unfortunately, the paper does not mention tool supports although empirical evaluation is shown. Therefore, the user has to manually perform the technique, or to develop its automated tool.

4.2. Change Impact Analysis for Software Revision in a software non-intensive project

In a paper\(^{23}\), software revision due to requirements changes is focused, and two techniques of estimating traceability between documents and source codes are proposed. The first technique is developed on the basis of answers of questionnaires from practitioners. The second one is developed on the basis of the feedback about the experiences of
the first one by the practitioners. Because software is not a central part in projects the technique will be applied, they cannot easily introduce software development methods such as software product line techniques and model driven development. According to our feature model in Figure 1, we characterized the first technique as follows.

- **Source**: Sections in requirements document.
- **Destination**: Functions in source codes.
- **Meaning**: Change impacts.
- **Assumption**: Users can specify suitable keywords and terms corresponding to a requirements change. Indirect impacts are out of scope.
- **Consequence**: Estimate. The quality features are as follows:
  - **Precision**: Yes.
  - **Recall**: Yes.
  - **Tradeoff**: Yes. Using the number of query keywords, the tradeoff is depicted in the same way in Figure 2. Data in actual industrial projects were used to calculate the metrics above. The detail of the projects were not published.
- **Preprocess**: Several samples of actual impacted codes should be manually identified.
- **Process**: Interactive. Several trials are required. During the trial, keywords may be updated. Pseudo-precision and recall are provided on the basis of samples defined in the preprocess so that the user can judge whether the keywords should be updated or not.
- **Tool**: integrated. The tool is developed as an independent application.

A model instance of the second technique is also as follows.

- **Source**: Sections in requirements document.
- **Destination**: Functions in source codes.
- **Meaning**: Change impacts.
- **Assumption**: Users can choose suitable keywords and terms corresponding to a requirements change from the candidates provided by the technique. The technique provides the candidates of the keywords.
- **Consequence**: Estimate. The quality features are as follows:
  - **Precision**: Yes.
  - **Recall**: Yes.
  - **Tradeoff**: No.
  Data in actual industrial projects were also used to calculate the metrics above. The detail of the projects were not also published.
- **Preprocess**: Several samples of actual impacted codes should be manually identified.
- **Process**: Interactive. Several trials are required. During the trial, keywords may be updated. Pseudo-precision and recall are provided on the basis of samples defined in the preprocess so that the user can judge whether the keywords should be updated or not. The technique recommends the candidates of new additional samples so that the number of samples increases.
- **Tool**: integrated. The tool is developed as an independent application.

The amount of user participation decreases when the technique is update from the first technique to second one. For example, the user simply chooses the keywords from candidates while he/she has to directly specify them in the first one. The second technique makes the number of samples for calculating pseudo-precision and recall increases so that the pseudo-metrics become more actual than ever.

In the techniques, the meaning of traceability relationships is clearly defined. Therefore, users can easily decide whether the techniques are suitable for them. In addition, the amount of the user participation is clearly stated. The potential users can also discuss whether they can accept additional efforts caused by the user participation.
4.3. MDD and Model Transformation

In a paper\(^{13}\), Model-driven development (MDD) and the traceability among the models are focused. In MDD, a model of high-level specification is systematically transformed into the lower-level models including source codes. The technique in this paper also generates trace models (traceability relationships) in the same way. The technique can be represented as follows according to our feature model.

- **Sources**: Model elements including source codes.
- **Destination**: The same as sources.
- **Assumption**: MDD is performed. Traceability meta-models and their transformation rules are predefined.
- **Consequence**: Determinate.
- **Process**: Batch.
- **Tool**: Integrated as an Eclipse plugin.

If MDD is fully used in projects of a user, there is some possibility of introducing this technique. However, several additional tasks are required in advance such as writing traceability meta-models. If the user accept such additional tasks, this technique is suitable for him/her.

5. Conclusion

In this paper, we proposed a feature model of traceability techniques. The model is designed so that the potential users of traceability techniques can choose a technique suitable for them. The model can be thus a basis of knowledge base for selecting suitable traceability techniques. Features of the model are chosen so that they can be obtained from published documents such as technical papers. Our feature model can be thus an answer of RQs mentioned in introduction. We also described model instances of several techniques on the basis of the feature model, and discussed which kinds of users were suitable for each technique.

We finally summarize the problems and future issues. Of course, our first future issue is to implement the knowledge base based on our feature model. The main challenge is to describe and register the instances of traceability techniques. In reality, each proposed technique in technical papers has its life span. For example, the tools of a technique proposed in old papers are not supported or opened to the public. We also take such life span into account when we develop the knowledge base. A unique reason for using traceability is the request by regulatory code or for certification. According to a survey\(^7\), about 40% of projects had such reason. Currently, we do not take such issue into account. Therefore, we have to examine the regulations and standards such as ISO 12207, ISO 26262 or SPICE. Our model is designed on the basis of papers published before September 2016\(^1\). We thus have to describe model instances of the techniques in recent papers, and the model should be revised if it should be. Finally, we want to evaluate our model. The simplest way of the evaluation is to ask several practitioners to choose a technique by using the knowledge base. Otherwise, we ask researchers investigate and examine the activities of practitioners to explore the suitable techniques for the practitioners. In such a case, we will be able to obtain more detailed feedback from the researchers.

References


